Título: TattooARt– Aplicación móvil de visualización de tatuajes con realidad aumentada.

Autora: Alejandra Jiménez López

Director: Pere-Pau Vázquez

Departamento: Dept. de Ciencias de la Computación

Titulación: Ingeniería en Informática

Centro: Facultad de Informática de Barcelona (FIB)

Universidad: Universitat Politècnica de Catalunya (UPC) BarcelonaTech

Agradecimientos

En primer lugar, me gustaría agradecer a Pere-Pau Vázquez por aceptar este proyecto tan distinto, por dejarme total libertad de decisión, por su atención, comprensión y cercanía.

En segundo lugar, agradezco a Pau Elías de la empresa Chloroplast Games, por encargarse del seguimiento de la planificación y desarrollo, y por estar siempre que lo he necesitado.

Por último a mi familia y amigos por mostrar su apoyo y colaborar en mis continuas pruebas durante el desarrollo

Resumen

Este proyecto se centra en el desarrollo de una aplicación móvil para *Android* llamada *TattooARt*. Está fundamentada en realidad aumentada y pretende cubrir una necesidad en un sector aún poco digitalizado: el del tatuaje.

El objetivo principal es conseguir que el cliente potencial del sector del tatuaje pueda, mediante su teléfono móvil, previsualizar en su piel el diseño que pretende tatuarse simplemente seleccionándolo de su galería propia o de una por defecto. Poder probar infinitas posibilidades previamente, no solo evitaría las malas decisiones, sino que supondría un gran ahorro en tiempo y material desechable al estudio o tatuador en cuestión.

Fundamentalmente, la aplicación consiste en un sistema AR para dispositivos móviles que mediante la detección de un patrón o *tracker*, que se colocará sobre la piel, sitúa el diseño seleccionado, adaptándolo a la curvatura del cuerpo y aplicando diferentes efectos gráficos para lograr el mayor realismo posible.

En el proyecto estudiaremos y determinaremos la mejor tecnología móvil de *AR (Augmented Reality)* que permita el buen funcionamiento de la aplicación. Además, necesitaremos adquirir conocimientos de la plataforma *Unity* y el lenguaje C# para progresar en su desarrollo.

Resum

Aquest projecte es centra en el desenvolupament d'una aplicació mòbil per a *Android* anomenada *TattooARt*. Està fonamentada en realitat augmentada i pretenent cobrir una necessitat en un sector poc digitalitzat: el tatuatge.

L'objectiu principal és aconseguir que el client potencial del sector del tatuatge pugui, mitjançant el seu telèfon mòbil, previsualitzar a la seva pell el disseny que pretengui tatuar-se: simplement seleccionant un per defecte o afegint-lo de la seva galeria propia.

Poder probar infinites possibiltats prèviament, no només implica evitar males decisions, sinó que suposa una gran estalvi en temps i material per a l'estudi o el tatuador en qüestió.

Fonamentalment, l'aplicació consta d'un sistema AR per a dispositius mòbils que mitjançant la detecció d'un patró o *tracker*, que es col·loca sobre la pell, es disposa el disseny seleccionat, adaptant-se a la curvatura del cos i aplicant diferents efectes per aconseguir el màxim realisme possible.

En el projecte estudiarem i determinarem la millor tecnologia mòbil d'AR (realitat augmentada) que permeti el millor funcionament de l'aplicació. A més, necessitarem adquirir coneixements de la plataforma Unity i el llenguatge C# per treballar en el seu desenvolupament.

Abstract

This project focuses on the development of a mobile application for Android called TattooARt. It is based on augmented reality and aims to cover a need in a sector poorly digitalized: the tattoo. The main objective is to ensure that the potential client of the tattoo sector can preview on his skin the design that is intended to be tattooed through his mobile phone: simply by selecting it from his own gallery or one by default. If the user is able to prove infinite possibilities beforehand, not only would avoid bad decisions, but It will be a great time saver for both the client and the tattoo artist.

Fundamentally, the application consists of an AR system for mobile devices that by detecting a pattern or tracker, which will be placed on the skin, places the selected design, adapting it to the curvature of the body and applying different graphic effects to achieve maximum realism as possible.

In the project we will study and determine the best mobile technology of AR (Augmented Reality) that allows the proper functioning of the application. In addition, we will need to acquire knowledge of the Unity platform and the C# language to progress in its development.

Índice de contenido

Índice de figuras

Índice de tablas

1 Introducción

1.1 Contexto

TattooARt será el nombre que recibirá este proyecto, título que liga los temas que lo definen: el tatuaje (tattoo), la realidad aumentada (AR) y el arte (art).

Claramente hay un impulso vocacional y profesional que me ha llevado a pensar en la necesidad de este proyecto, unir dos mundos que me apasionan y forman parte, en mayor o menor medida, de mi vida profesional.

Hablamos en primer lugar del arte del **tatuaje**, un arte muy antiguo en el que cada vez más gente sitúa su punto de mira, tanto por tradición, tendencia, situación emocional, curiosidad, vocación o interés económico. Y es que para entender este fenómeno, que algunos consideran mera tendencia, hay que remontarse al neolítico, donde encontraron los primeros restos de un ser humano tatuado. Desde entonces generaciones inmemoriales han utilizado esta técnica ancestral, dotándola de significados positivos y negativos, evolucionando técnicas y mejorando las herramientas. Pues bien, tras miles de años de historia, una máquina de tatuar llegó a mi mano a los 16 años, empecé el camino del aprendizaje y lo continúo actualmente, 6 años más tarde, siendo ahora mi profesión y vocación.

Hablamos en segundo lugar de la **realidad aumentada o AR**, que engloba el conjunto de tecnologías que permiten que el usuario visualice una parte de la realidad a través de un dispositivo tecnológico añadiendo información gráfica. Sus orígenes son mucho más recientes, en 1962 se construyó un prototipo llamado *Sensorama* que nos invitaba a la primera experiencia sensorial de realidad aumentada. La obsesión con lo visual, con el diseño 3D y los **gráficos**, me acompaña ya desde la primera película infantil que vi y me motivó para enfocar mis estudios hacia las ciencias de la computación.

¿Cómo podríamos unir dos conceptos tan lejanos? Aparentemente son dos ámbitos paralelos y distantes, pero no se juntan en el infinito, porque en estos últimos años ya se han encontrado y parece que se van a acompañar indefinidamente.

La idea del proyecto surgió años atrás a partir de una **problemática** que, como tatuadora, perjudicaba la eficiencia y retrasaba la ejecución del trabajo: el cliente se presenta con el diseño que quiere, o con el que ha sido diseñado para él; entonces empieza un proceso en el que elige en qué tamaño y posición tatuarselo; se aplica un calco bastante resistente a ser borrado y si está todo bien se procede a realizar el tatuaje. No siempre está todo bien, no

siempre es el tamaño acertado, ni la posición correcta, ni es como el cliente lo imaginaba. Entonces toca borrar, irritar la piel, y repetir este proceso hasta que el usuario quede satisfecho.

TattooARt será una aplicación móvil que se encargará de sustituir esta fase. Una aplicación móvil es una solución informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles. Las aplicaciones permiten al usuario efectuar un conjunto de tareas de cualquier tipo. En este caso, mediante las tecnologías de realidad aumentada, cubriría la principal funcionalidad de poder visualizar una simulación del tatuaje en vivo a través de la pantalla de un dispositivo móvil, en el tamaño y posición deseada, admitiendo rotación y escalado e infinitas posibilidades antes de hacerlo irreversible.

A grandes trazos, la aplicación móvil consistiría en un sistema de realidad aumentada que a través de la cámara del dispositivo detecta y sigue un marcador (un patrón que sirve para fijar en una posición del mundo real lo que se quiere superponer) y en el lugar del marcador superpone la información sintética, en nuestro caso el tatuaje. Por esta razón, el nombre de la técnica: se "aumenta" la realidad con información digital.

En un primer lugar, planteábamos la idea de dibujar el patrón en la piel con un rotulador. En consecuencia de limitaciones tecnológicas que nos encontraremos más adelante, la idea inicial tendrá que ser remodelada y en esta primera versión de la aplicación se necesitará un adhesivo del patrón con el logo de la aplicación, potenciando así la actitud de la marca y la consistencia de la app.

Está claro que el **público objetivo** no está formado únicamente por tatuadores, el propio cliente puede probar en casa modelos innumerables y decidirse por uno u otro, o simplemente jugar por curiosidad. No olvidemos que alrededor de esta práctica se alimenta negocio bastante sólido y por tanto es una apuesta fiable trabajar en la mejora de sus herramientas aprovechando las carencias informáticas de las que dispone.

1.2 Justificación

Actualmente están a nuestro alcance sistemas cuya tecnología AR nos sería útil para esta aplicación, los ejemplos más conocidos y claros son Instagram o Snapchat, entre otros. Hay algunas aplicaciones que como tal intentan cubrir la necesidad específica que se plantea, pero no cumplen las expectativas.

Destacamos entre ellas INKHUNTER, una aplicación que permite subir diseños de una galería de artistas o tuyos propios, dibujarte un marcador en la piel que debe detectar la cámara de tu

dispositivo móvil y entonces coloca el diseño elegido en un tamaño predeterminado. Puedes tomar una fotografía y entonces variar la posición del diseño.
TattooARt aspira a ser una idea mejorada de este estilo de aplicaciones. El usuario podrá colocar el diseño al tamaño y posición deseada previamente a realizar una fotografía, podrá grabar, una vez ajustado, sus movimientos naturales para poder previsualizar cómo quedaría en la realidad.
No nos serviría el sistema utilizado en todas las aplicaciones enfocadas a este ámbito porque no se preocupan de la adaptabilidad al cuerpo donde se proyecta, la curvatura, la luz y el color. Por lo tanto destacamos la necesidad de crear una nueva solución informática debido a la falta de calidad de las aplicaciones que compiten en este mercado.

1.3 Alcance

El principal objetivo de este proyecto es simular mediante un app de AR la aplicación de un tatuaje al usuario, ajustándose al máximo a la realidad.
Para esto debemos desglosar ciertos sub-objetivos que deberían dar con el principal.
En primer lugar deberíamos decidir qué tecnología nos va a ser de más ayuda, qué funcionalidades tiene cada una y cuál nos generará menos problemas de cara al objetivo principal.
En segundo lugar, deberíamos elegir el método y nivel de realidad aumentada que necesitaremos, cuál nos va a generar menos vibraciones y cuál nos va a dar un mejor resultado según el método elegido (detección de superficies, o marcadores, entre otros).
Por último, deberíamos dar con el método de fusión entre todas las opciones de *blending* de imágenes que fuese más fiel al resultado real de un tatuaje en la piel.
Como funcionalidades principales destacamos:

- Galería de diseños
- Buscador en galería
- Opción para subir diseños directamente desde el dispositivo
- Función para capturar una vez el *marker* se ha reconocido
- Guardar las capturas en la galería del usuario

Obviamente encontraremos ciertas dificultades para lograr este objetivo. Entre las cosas que se pueden considerar obstáculos en menor o mayor repercusión pueden ser:

- Problemas con la tecnología AR elegida
- Problemas con las características del patrón mínimo

- Técnicas de percepción que generen temblor
- Capacidad de los dispositivos móviles respecto AR, así como de los *shaders* del tatuaje y sus modos de fusión
- Problemas con la adaptabilidad a la deformación en zonas muy curvadas o con superficies corporales de aplicación más exigentes
- Gestión del detector de marcadores para tonos de piel más oscuros

1.4 Metodología y rigor

En este apartado es dónde deberían debatirse: las decisiones técnicas para el desarrollo del sistema TattooARt, la arquitectura del sistema, las metodologías de programación y los patrones de diseño.

Para que el proyecto siga la trayectoria esperada es muy importante seguir firmemente las pautas de diseño, arquitectura y metodologías que te aseguren el desarrollo de un proyecto sólido y consistente.

En una planificación general del desarrollo a priori, identificamos unas fases claras:

- Análisis: La primera de ellas es la de análisis y documentación, con la que haremos pruebas para resolver los riesgos y obstáculos. Estudiaremos las tecnologías y su funcionamiento.
- Planificación: debemos elegir el patrón de diseño, la arquitectura y la metodología de trabajo y si realizaremos una aplicación nativa o mediante una plataforma de desarrollo.
- Desarrollo: a partir de la tecnología escogida se empieza con el desarrollo de la app.
- Seguimiento y testeo: realizaremos test de funcionalidad en diferentes situaciones alternativas y encargaremos la prueba de la aplicación a diferentes usuarios estándar para conocer su *feedback.*

Las primeras pruebas se realizaron con ARKit en el entorno de iOs y observamos buena arquitectura de detección del tracker. Como no disponíamos de un dispositivo como iPhone ni iPad, ni tampoco de un Mac para generar las compilaciones, decidimos hacerlo con las herramientas de ARCore y la plataforma de Unity. Para nuestra sorpresa y desesperación, como veremos más adelante, se encontraron diferencias funcionales que dependían de la implementación interna de ARKit y ARCore que no nos permitieron seguir con la idea original. Eso supuso un giro en la trayectoria del trabajo, un cambio pequeño al que costó mucho llegar pero que se afrontó con ilusión y optimismo al analizar las nuevas ventajas.

2 Planificación

2.1 Datos globales

Para determinar una planificación temporal general del proyecto hace falta establecer una fecha de inicio y una fecha de finalización. En nuestro caso, ciñéndose al turno de lectura, que es del jueves 23 enero 2020 a miércoles 29 enero 2020, estableceremos una fecha de inicio en el 01/10/2019 y la finalización en el 01/01/2020. Según las tareas que hemos determinado y su coste temporal total de 200 horas, marcamos necesaria una dedicación diaria de 4 horas en un total de 50 días laborales.

2.2 Gestión de las tareas

2.2.1 Identificación y descripción de las tareas

1. **Gestión del proyecto:** agrupa todas las tareas de gestión, así como las reuniones, la documentación, el alcance, la planificación, el cálculo del presupuesto, la redacción del informe de sostenibilidad y compromiso social, y recogida de las referencias.
2. **Análisis de la competencia:** buscar en App store y Play store aplicaciones que pueden cubrir funcionalidades similares a la que propone el proyecto. Si hay una destacablemente mejor desarrollada y con mejor aceptación en el mercado, basarse en esta para realizarle un estudio analítico.
3. **Comparativa con la app destacada**: una vez probamos la app que consideramos más competente en diferentes casos de uso, toca decidir qué funcionalidades queremos mantener, descartar o mejorar, y qué alternativas proponemos a algunas funciones.
4. **Definición funcional:** Antes de realizar los *wireframes* en el desarrollo de aplicaciones, podremos definir primero el funcionamiento de la *app* sin ningún tipo de diseño nos permitirá establecer una navegación sencilla, funcional y detectar fácilmente los errores que puedan haber.
5. **Diseño del esquema de la aplicación:** Determinar qué funcionalidades queremos que tenga la aplicación, así como el diseño de los *wireframes*.
6. **Pruebas con diferentes herramientas:** es importante conocer qué herramientas y facilidades nos ofrecen las diferentes plataformas, si hemos decidido no hacer la aplicación nativa, esta elección será primordial para el progreso del proyecto.

7. **Pruebas con las tecnologías AR:** debemos analizar qué limitaciones y facilidades nos conceden las tres distintas tecnologías más grandes de AR: ARKit o ARCore. Y decidirnos por una de ellas.
8. **Decidir la plataforma:** antes de empezar a desarrollar deberemos decidir si lanzaremos la app para Android o iOS.
9. **Nivel de AR y diseño del patrón:** sabemos que la realidad aumentada se divide en niveles, nos centraremos en el nivel 1, en el que las aplicaciones utilizan marcadores habitualmente para el reconocimiento de patrones, el cual deberemos diseñar y testear.
10. **Diseño:** diseñaremos los *assets* del contenido y cómo será visualmente la interfaz.
11. **Programación del código:** en esta fase toca crear la estructura sobre la cual se apoyará el funcionamiento de la aplicación.
12. **Testeo:** con una versión beta deberemos realizar tests, también observar el comportamiento de usuarios aleatorios y familiares que puedan probarla en nuestra presencia. Otra manera de testear es desarrollar un *User Acceptance Test* (UAT), que se puede hacer en un *Excel*, dividiendo todas las tareas de la app en tareas sencillas, línea a línea e ir haciendo un recorrido por toda la *app*, con el resultado esperado y el obtenido.
13. **Corrección de bugs:** Una vez que existe la versión inicial, dedicaremos gran parte del tiempo a corregir errores funcionales para asegurar el correcto funcionamiento.
14. **Mejoras:** sabiendo qué cosas están fallando de la versión final tras el testeo solo hace falta mejorarlas y volver a testear.
15. **Herramientas de seguimiento:** se puede implementar la API de *Google Analytics* para saber multitud datos, entre ellos, información acerca del *hardware* como el modelo del móvil o tamaño pantalla, el sistema operativo, el país del usuario...etc
    A partir de aquí podríamos monitorizar cuánta gente utiliza un determinado botón de la interfaz o el tiempo que invierten en el uso de la *app*, etc.
16. **Publicación:** crearemos una cuenta de desarrollador y lo subiremos a la plataforma
17. **Seguimiento:** se realiza un seguimiento a través de analíticas, estadísticas y comentarios de usuarios, para evaluar el comportamiento y desempeño de la *app*.
18. **Actualización:** corregir errores, realizar mejoras y actualizarla en futuras versiones.

2.2.2 Tabla de tareas

A continuación mostramos una tabla con las tareas etiquetadas alfabéticamente, con el tiempo estimado que se plantea dedicar y las dependencias entre ellas.

	Descripción	Tiempo estimado	Dependencias
A	**Gestión del proyecto**	**40h**	
B	**Análisis de la competencia**	**4h**	
C	**Comparativa con la app destacada**	**4h**	**B**
D	**Definición funcional**	**4h**	**C**
E	**Diseño del esquema de la aplicación**	**8h**	**D**
F	**Pruebas con diferentes herramientas**	**8h**	
G	**Pruebas con las tecnologías AR**	**8h**	**F**
H	**Decidir la plataforma**	**4h**	
I	**Nivel de AR y diseño del patrón**	**8h**	
J	**Diseño**	**16h**	
K	**Programación del código**	**32h**	**J**
L	**Testeo**	**12h**	**K**
M	**Corrección de bugs**	**8h**	**L**
N	**Mejoras**	**12h**	**M**
O	**Herramientas de seguimiento**	**8h**	**N**
P	**Publicación**	**4h**	**O**
Q	**Seguimiento**	**8h**	**P**
R	**Actualización**	**12h**	**Q**
Total de horas		**200h**	

Tabla 1: Tabla de tareas

*Si M contiene a L en dependencias, M debe finalizarse antes que L.

Además debemos considerar los recursos humanos y materiales de los que dispondremos. En primer lugar necesitaremos contar con un espacio donde trabajar y sus correspondientes instalaciones y servicios.

En cuanto a material, necesitaremos útiles de oficina, un portátil *Asus*, un móvil *Huawei* p20, y un *router* inalámbrico.

En segundo lugar necesitaremos tres perfiles que completen las 200 horas aproximadas del proyecto: un analista, un diseñador y un desarrollador.
Más adelante en el apartado de presupuesto y sostenibilidad, cuantificaremos estos recursos de una forma más ajustada y realista.

2.3 Planificación temporal

2.3.1 Diagrama de *Gantt* i calendario

A continuación, mostramos un diagrama de *Gantt* donde cada bloque de color incluye la cantidad de horas que supone esa tarea y su tamaño la continuidad en el tiempo:

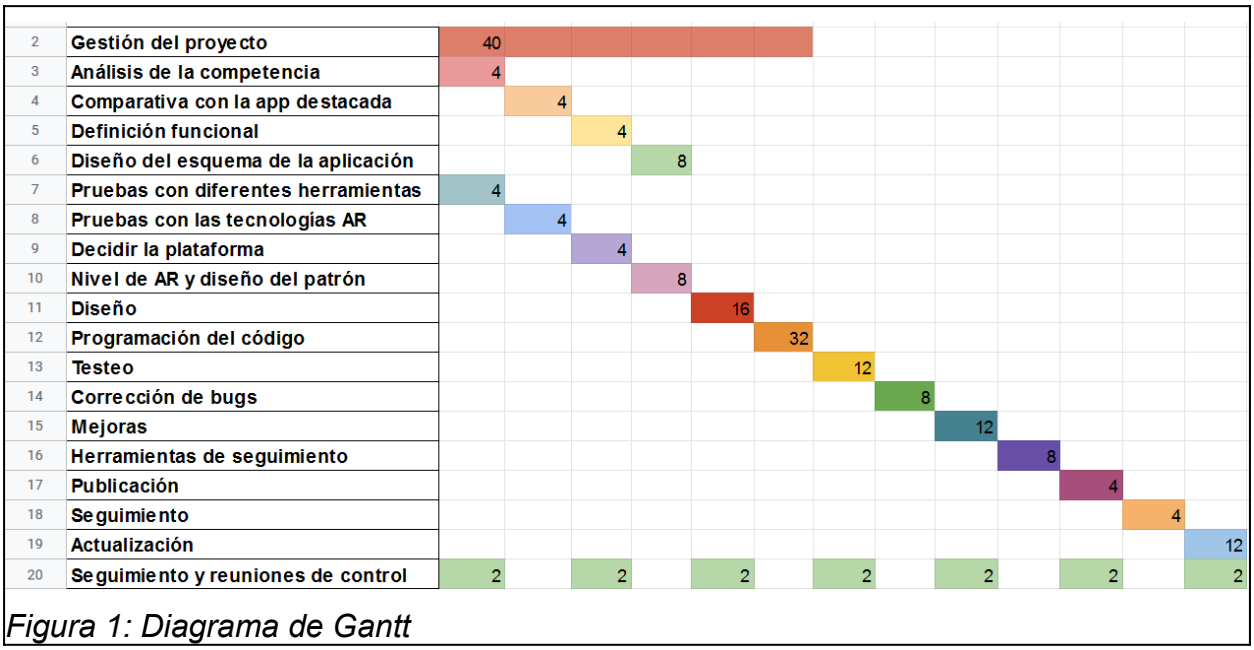

Figura 1: Diagrama de Gantt

Como hemos definido 50 días como tiempo aproximado de trabajo, ajustaremos las tareas a 10 semanas de 5 días laborables.

Semanas	Intervalo temporal	Tareas a trabajar
Semana 1	7-11 oct	**A, B, C, D, E**
Semana 2	14-18 oct	**A, E, F, G**
Semana 3	21-25 oct	**A, H, I**
Semana 4	28-1 nov	**J, K**
Semana 5	4-8 nov	**K**
Semana 6	11-15 nov	**K, L**

Semana 7	18-22 nov	**L, M**
Semana 8	25-29 nov	**N, O**
Semana 9	2-6 dic	**O, P, Q**
Semana 10	9-13 dic	**Q, R**

Tabla 2: Calendario de tareas

2.3.2 Revisión de la planificación temporal

Los problemas con las tecnologías implicadas nos ha supuesto un desplazamiento temporal de las tareas asignadas con una carga de 2 semanas de retraso. La solución a esta demora es aumentar la carga de trabajo diario en los días restantes de la planificación e intentar reducir el impacto cualitativo de la presentación final del trabajo. Las tareas que han ralentizado el desarrollo han sido:

- **G. Pruebas con las tecnologías AR:** analizar qué limitaciones y facilidades nos conceden las tres distintas tecnologías más grandes de AR: ARKit, ARCore y Vuforia. En esta tarea triplicamos el tiempo previsto. Inicialmente realizamos pruebas con ARKit, que nos permitían lograr el objetivo inicial de la app y basarla en un patrón simple dibujado a mano alzada. El proyecto planeábamos hacerlo con ARCore compatible con plataformas de desarrollo y testeo Android, por disponibilidad material. Fue entonces cuando descubrimos que no compartían la misma arquitectura interna y que el patrón debía ser más complejo e idéntico al original en todos sus test. La solución con la que finalizamos esta tarea la veremos más adelante en la gestión de riesgos y alternativas.
- **I. Nivel de AR y diseño del patrón:** desde el primer momento supimos que en el nivel de AR que podíamos defender era el que funcionaba con el reconocimiento del patrón. Fue el diseño de este patrón el que nos llevó más tiempo. Tuvimos que servirnos de una herramienta llamada *arcoreimg tool* que a partir de una imagen generaba una puntuación del 0 al 100, siendo la máxima puntuación la de un patrón que podía funcionar pero que no era totalmente fiable, y debíamos mejorar analizando qué características sumaban puntuación a creación de una buena imagen de referencia.

A continuación mostramos la tabla actualizada incluyendo la demora temporal que han supuesto estas dos tareas:

Semanas	Intervalo temporal	Tareas a trabajar
Semana 1	7-11 oct	**A, B, C, D, E**

Semana 2	14-18 oct	**A, E, F**
Semana 3	21-25 oct	**A, G, H, I**
Semana 4	28-1 nov	**G, I**
Semana 5	4-8 nov	**G, I**
Semana 6	11-15 nov	**J, K**
Semana 7	18-22 nov	**K**
Semana 8	25-29 nov	**K, L**
Semana 9	2-6 dic	**L, M**
Semana 10	9-13 dic	**N, O**
Semana 11	16-20 dic	**O, P, Q**
Semana 12	23-27 dic	**Q, R**

Tabla 3: Calendario de tareas revisado

2.4 Gestión del riesgo

A continuación mostramos en negrita las adversidades que deberíamos evitar y cómo vamos a gestionar ese riesgo, valorando posibles planes solución:

- **Problemas con la tecnología AR elegida:** si a pesar de haber realizado un testeo e investigación previa a la elección de la tecnología AR, una vez empezado el desarrollo con esta tecnología encontramos lagunas o errores que no nos permitan avanzar, deberemos cambiar a otra de las consideradas, analizando previamente que cubra las necesidades que nos faltaban.
- **Evitar técnicas de percepción que generen temblor**: para evitar el temblor si es que nos encontramos con este problema tan común en AR, deberemos disminuir la carga gráfica de la aplicación, reduciremos los objetos pesados y optaremos por la elección de shaders más simples.
- **Capacidad de los dispositivos móviles respecto AR:** planteamos dos soluciones, o bien trabajar con móviles más potentes o adaptar la aplicación a un público de dispositivos más sencillos. Como queremos lograr una aplicación de calidad nos decantaremos por la primera opción.
- ***Shaders* del tatuaje y sus modos de fusión:** en este caso para solucionar esta adversidad que juega un papel muy importante en el realismo y el resultado final de

la proyección decidimos hacer un conjunto de pruebas con diferentes *shaders* y modos de fusión en diferentes casos de uso. Una tabla comparativa de imágenes nos ayudará a elegir la mejor combinación

- **Problemas con la adaptabilidad a la deformación en zonas muy curvadas o con superficies corporales de aplicación más exigentes:** este es realmente el problema de mayor envergadura, aunque aceptamos que hay ciertas exigencias a las que una aplicación móvil no alcanza, adaptar el tatuaje a zonas de curvaturas muy complejas deberá solucionarse con un diseño del patrón que de mucha información al sistema de AR sobre la deformación de la zona.
- **Gestión del detector de marcadores para tonos de piel más oscuros:** en este caso planteamos la opción de identificar patrones con los colores invertidos, de manera que deberíamos añadir también esta explicación en los wireframes aclaratorios del uso de app.

2.4.1 Obstáculos

El primer punto del apartado de gestión de riesgo relacionado con los **problemas con la tecnología AR elegida** ha sido el mayor obstáculo con el que nos hemos encontrado.

Inicialmente, la idea esencial de la app consistía en definir un patrón de reconocimiento que al pintarlo con un rotulador negro o azul en la piel, fuera detectado por la cámara de AR y permitiera adaptar con realidad aumentada un diseño de una galería por defecto o de una propia.

Esta idea base la probamos con ARKit, un *iPhone* y un *Mac*, del que disponíamos solo para la realización de la prueba. Y vimos que funcionaba a la perfección, así que nos lanzamos a desarrollarlo con el material a nuestro alcance y el que formaba parte del presupuesto inicial: *ARCore*, un *Huawei* P20 y un portátil *ASUS*.

El obstáculo se produjo cuando vimos que *ARCore* necesita un patrón muy complejo para generar las suficientes *features*[1], y no podíamos dar con un patrón mínimo que fuese lo suficientemente sencillo. Pues si la intención era que cualquier usuario lo pudiera pintar en la piel, un patrón complejo supondría una labor tediosa para cualquiera.

Además, no solo necesitaba muchas características para generar una buena imagen de referencia si no que, no admitía la mínima variabilidad con el patrón original, cosa que implicaba que una reproducción acertada en rotulador tampoco sería reconocida por la tecnología de ARCore.

[1]Elementos que se detectan en la imagen y que sirven para que el sistema sepa dónde está el patrón y pueda hacerle un seguimiento aunque éste se mueva.

2.4.2 Búsqueda de alternativas y soluciones

Llegados a este punto en el que identificamos las limitaciones de la tecnología escogida y no podíamos seguir con la idea original, planteamos diferentes alternativas:

- Cambiar el material de desarrollo: suponía disponer de un *iPhone* o *iPad* de las últimas generaciones para poder beneficiarse de su tecnología AR, y un *Mac* para compilar o una máquina virtual.
- Cambiar la idea original: si la metodología de funcionamiento de la app no era la deseada, nos tocaba ceder a alguna otra manera de impresión del patrón.

A favor de la primera alternativa, encontramos que era una buena inversión poder disfrutar de las posibilidades profesionales que concede *Apple*, siempre elegida por diseñadores. Pero los factores en contra son determinantes: supone un aumento en el presupuesto que no podemos asumir, un desconocimiento del entorno de *iOs*, y la inseguridad de poder llegar a compilar el proyecto satisfactoriamente en una *Máquina Virtual*.

Analizando la segunda alternativa, en un primer lugar rechazamos la idea que el cliente tuviera que imprimir un patrón que pudiera descargar de la *app* y adaptarlo con más o menos éxito y con las herramientas caseras adhesivas de las que dispusiera. Esto restaría mucha usabilidad a la aplicación porque supone casi el mismo esfuerzo que imprimir el tatuaje y probarlo directamente en la piel. Entonces hemos desarrollado la idea de fabricar unas pegatinas con un logo de la *app* y fondo transparente. Esto tiene algunas puntos en contra:

- No todo usuario puede disponer de estas pegatinas. Únicamente a quien se le haga llegar.
- La caducidad de estas, ya que con uno o dos usos perderían la adhesividad.
- Pero a favor de esta solución encontramos otras ventajas que la harían la propuesta definitiva para la primera versión de la *app*:
- El hecho de repartir pegatinas a los clientes que vengan a tatuarse o informarse genera una identidad y personalidad a la marca y relacionarán así el uso de esta aplicación innovadora con el artista tatuador en cuestión.
- Nos permitirá repartir las pegatinas a modo de tarjeta de visita, con información de contacto con el estudio incluso de enlazar la identificación del patrón con un enlace virtual a la web del artista.
- Evitará que el cliente tenga que mancharse la piel para probar la aplicación con la idea que desea.

Así que, más adelante veremos como diseñaremos un logo para la *app* y encargaremos la primera tanda de adhesivos para poder comprobar su buen funcionamiento.

3 Conocimientos previos

3.1 Tatuaje

3.1.1 Definición

Un tatuaje es una forma de decoración corporal en la que un diseño se reproduce en la piel insertando diferentes tintas, ya sean permanentes o temporales, en la capa de la dermis de la piel para cambiar su pigmento. Los tatuajes se diferencian en muchas categorías, como pueden ser puramente decorativos, simbólicos, pictóricos, médicos, identificativos o religiosos, entre otros.

3.1.2 Etimología

La palabra *tatuaje* nació en el siglo XVIII como préstamo de la palabra samoana *tatau*, que significa "golpear". En cambio, la palabra inglesa *tattoo* se deriva de la palabra holandesa taptoe. Antes de la importación de la palabra polinesia, la práctica del tatuaje había sido descrita en Occidente como pintura, cicatrización o tinción.

En japón disponen de la palabra japonesa *irezumi* significa "inserción de tinta" que se refiere a los tatuajes realizados mediante el método tradicional japonés de inserción manual. La palabra más común utilizada para los diseños de tatuajes japoneses tradicionales es *horimono*, y en cambio utilizan la palabra *tatuaje* para referirse a los estilos de tatuaje no japoneses. El antropólogo británico Ling Roth en 1900 describió cuatro métodos de marcado de la piel y sugirió que se diferenciaran bajo los nombres *"tatu", "moko", "cicatrix"* y *"queloide"*

3.1.3 Proceso de trabajo previo al tatuaje

La realización de un tatuaje conlleva un proceso que se sigue de manera general.

1. En primer lugar el tatuador diseña el tatuaje a partir de las ideas que el cliente le ha explicado y se lo muestra al cliente. Si está conforme y convencido sobre la pieza diseñada, empieza el proceso manual de preparación de la plantilla.
2. El dibujo plantilla se confecciona realizando un trazado de lineas que delimita las sombras y líneas de referencia del tatuaje.

3. Una vez realizado se imprime con una impresora común en diferentes tamaños, y se elige uno de ellos orientativamente según la zona del cuerpo donde quiere tatuarse.
4. Entonces se realiza una impresión térmica o manual, sobre un papel hectográfico especial.
5. Se extiende sobre la piel una crema específica para la transferencia de la calca y se adhiere el papel vegetal con la plantilla. Se espera 5-10 minutos y se retira el papel.

En este punto, el cliente puede comprobar si el tamaño y la posición es la adecuada, si no es así, se debe empezar el proceso desde el punto 3 imprimiendo el diseño de nuevo.

Una vez está convencido, se espera unos 5-10 minutos a que seque la plantilla y empieza el proceso de inyección de la tinta según el método que utilice dicho tatuador.

Figura 2: Diseño de un tatuaje

Figura 3: Plantilla de un tatuaje

Figura 4: Simulación de plantilla sobre la piel

Figura 5: Tatuaje finalizado

3.1.4 Relativo al proyecto

Esta aplicación generaría una mejora y aceleración del proceso anterior, dado que el cliente puede previsualizar el diseño de una manera mucho más fiel que la plantilla, puede dar con el tamaño y la posición mucho más rápido y evitar así la repetición del proceso.

Además puede probar múltiples opciones y lugares distintos sin ningún tipo de inversión material ni temporal, simplemente colocando el patrón en diferentes zonas. Hemos añadido una información en la pantalla sobre el tamaño al que se está escalando el plano. Esto último es muy útil para el tatuador que debe imprimir la plantilla, pues así no hay margen de error:

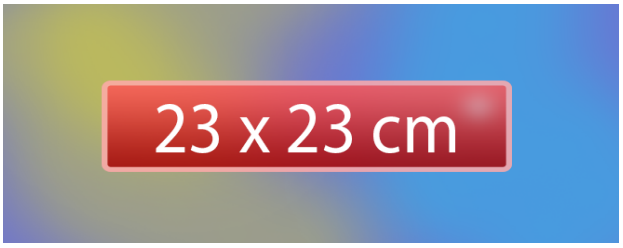

Figura 6: Información sobre el tamaño del diseño mostrada en la pantalla

Esto generaría el siguiente ahorro estimado:

- **Temporal:** de 30 minutos a 1 hora.
- **Económico:** de 8 a 10 euros, en material de impresión, papel hectográfico y cosméticos de transferencia y limpieza.

Si tenemos en cuenta el proceso anterior, esta aplicación añadiría un paso entre el 1 y el 2 que ahorraría tiempo en los pasos que le siguen. Mostramos una imagen de cómo sería este paso gracias a la aplicación móvil:

Figura 7: Paso previo a la aplicación de la plantilla

3.2 AR

3.2.1 Definición

AR es un entorno compatible con objetos reales y virtuales representados en tiempo real, es decir, incluye tanto la realidad virtual como elementos del mundo real.

El objetivo del sistema AR es agregar información y mejorar la experiencia del usuario en el entorno real. Esta definición de sistema AR no se limita a las tecnologías de visualización, sino que se puede aplicar al oído, el olfato y al tacto. AR es una parte de la realidad mixta. El entorno de virtualidad aumentada y el entorno virtual (VR) son virtuales o ficticios, mientras que el entorno de AR es real.

La estudio de la realidad aumentada incluye, entre otros conceptos, la detección y seguimiento de marcadores, creación de características, detección y seguimiento de movimiento, análisis de imágenes, reconocimiento de gestos y construcción de entornos controlados con varios sensores diferentes.

La realidad aumentada ha generado enormes cambios en el mercado actual. En dimensiones completamente innovadoras, las compañías de desarrollo de aplicaciones móviles van reduciendo la brecha entre la imaginación y la realidad.

3.2.2 Terminología

Fue en 1992 cuando, el investigador de la fábrica de aviones *Boeing*, Tom Caudell inventó el término realidad aumentada. En ese momento era una pantalla digital en la cabeza que servía de guía a los trabajadores de la fábrica para poder entender la disposición del montaje de los tan liosos cables. En esta primera definición de realidad aumentada se refería al conjunto elementos virtuales que se mezclaban con el mundo real para mejorar la percepción del usuario.

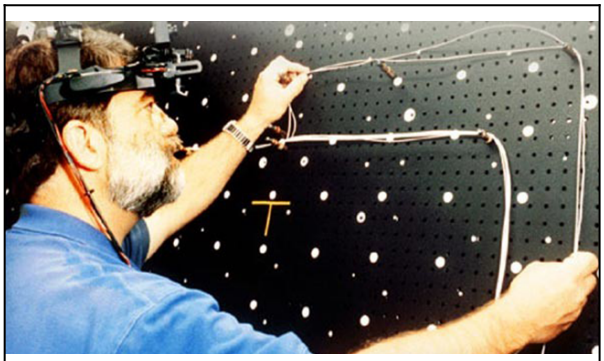

Figura 8: Tom Caudell en Boeing

Más adelante, en 1994, Paul Milgram presentó el esquema de la continuidad entre realidad y virtualidad, definiendo el nuevo término de la realidad mixta. Un sistema de realidad mixta fusiona el mundo real y el mundo virtual para producir un nuevo entorno donde los objetos físicos y digitales coexisten e interactúan. La realidad en este contexto se refiere al entorno físico y visible, como se ve directamente o a través de una pantalla de video.

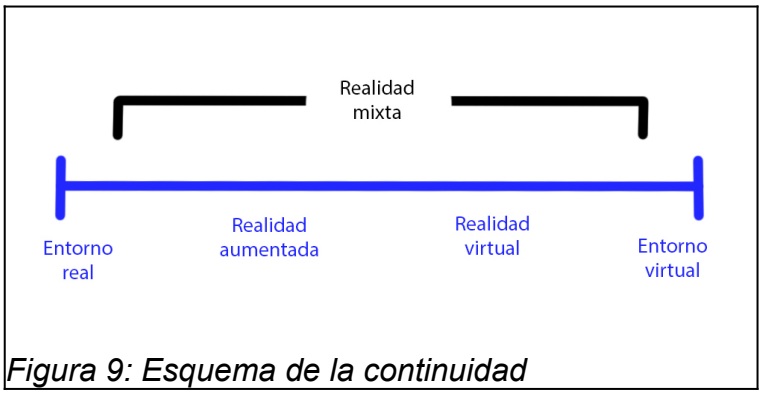

Figura 9: Esquema de la continuidad

3.2.3 Niveles de AR

Los teléfonos inteligentes son probablemente el medio más cercano para que usuario común pueda disfrutar de la experiencia de la realidad aumentada. Aún así no son el único medio para poder desarrollarla. A continuación veremos una pequeña introducción a los diferentes tipos de realidad aumentada que coexisten en este momento; cabe decir que hay algunos otros que no pueden clasificarse fácilmente entre uno de los mencionados seguidamente. Como cualquier campo de investigación, está en continua evolución y desarrollo, así que se debe considerar la posibilidad de cambio y diversificación de los tipos descritos.

3.2.3.1 Realidad aumentada basada en marcadores

También conocido como Reconocimiento de imagen o AR basado en reconocimiento. Este tipo de AR nos proporciona más información sobre el objeto después de haberlo reconocido. La tecnología AR basada en marcadores tiene muchos usos según los propósitos del mercado, como los simples y conocidos *códigos QR*.

La cámara reconoce el marcador, y lo reemplaza en la pantalla con una versión 3D del objeto correspondiente. El usuario puede ver el objeto desde varios ángulos. Además, mientras gira el marcador, el usuario también puede girar las imágenes en 3D, es decir, el marcador actúa como referencia para la aplicación AR que se ejecuta en el sistema.

Figura 10: Ejemplo de AR basado en marcadores

3.2.3.2 Realidad aumentada sin marcadores

La realidad aumentada sin marcadores es una de las aplicaciones de AR más implementadas en la industria.

También se conoce como AR basada en la ubicación, y funciona gracias a las tecnologías y funcionalidades de los nuevos dispositivos móviles inteligentes que proporcionan detección de la ubicación suficientemente precisa para muchas aplicaciones. Este método funciona leyendo datos del GPS, de conexiones Wi-Fi, de la brújula digital y el acelerómetro del dispositivo móvil, prediciendo, así, dónde está enfocando el usuario. Este modo de AR consiste generalmente en agregar información de ubicación en la pantalla sobre los objetos que se están viendo desde la cámara del usuario.

Figura 11: Ejemplo de AR sin marcadores

3.2.3.3 Realidad aumentada con proyección

Este es una de los tipos de AR más simples, consiste la proyección de luz en una superficie. El tipo de AR basado en proyección es interactivo, la luz se proyecta sobre una superficie y la interacción se realiza tocando la superficie proyectada con la mano. Los usos más comunes de las técnicas de AR basadas en la proyección se pueden utilizar para crear engaños sobre la posición, orientación y profundidad de un objeto. Esta tecnología ofrece muchas más posibilidades para experimentar y avanzar en el inminente sector de la Realidad Aumentada.

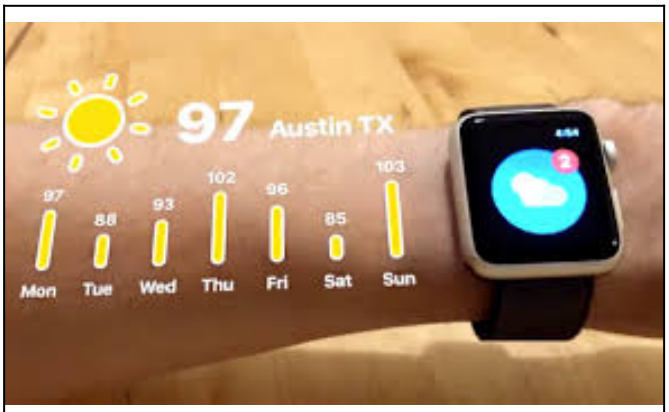

Figura 12: Ejemplo de realidad aumentada con proyección

3.2.3.4 Superposición basada en realidad aumentada

El AR basado en superposición proporciona una vista "alternativa" del objeto en cuestión, ya sea reemplazando toda la vista con una vista aumentada del objeto o reemplazando una parte de la vista del objeto con una vista aumentada. El reconocimiento de objetos juega un papel vital: si la aplicación no sabe lo que está mirando, no puede reemplazar la vista original por una vista aumentada.

Figura 13: Ejemplo de superposición basado en realidad aumentada

3.2.4 Relativo al proyecto

En este proyecto nos serviremos de un sistema simple de realidad aumentada que consiste en una cámara, una unidad computacional en el móvil y una pantalla. La cámara captura una imagen, y luego el sistema añade los objetos virtuales y muestra el resultado conjunto.
A partir del patrón de referencia el sistema AR calcula la pose relativa de la cámara en tiempo real. El término pose significa la posición con seis grados de libertad, es decir, la ubicación 3D y la orientación 3D de un objeto. El módulo de AR le permite al sistema agregar componentes virtuales a la escena real. Los objetos virtuales se mueven y giran en coordenadas 3D en lugar de coordenadas de imágenes 2D, y esta es la diferencia fundamental entre la realidad aumentada y otras herramientas de procesamiento de imágenes.

4 Tecnologías implicadas

4.1 Plataforma de desarrollo

Antes de pensar en la tecnología de AR deberíamos decidir qué plataforma nos servirá como herramienta principal para poder desplegar el proyecto.

Como hay una motivación previa por aprender el funcionamiento de Unity 3D y por todas las facilidades que nos brinda este sistema multiplataforma, no hemos invertido demasiado tiempo en valorar otras alternativas.

Por lo tanto, para desarrollar *TattooARt* hemos elegido *Unity*, un motor de videojuego multiplataforma creado por *Unity Technologies* gratuito en la versión personal.

El enfoque de esta compañía es "democratizar el desarrollo de juegos y aplicaciones", y hacer lo más accesible posible al público el desarrollo de contenidos interactivos en 2D y 3D.

Es una plataforma sencilla para desarrolladores noveles al igual que muy completa para profesionales. Unity está disponible para *Microsoft Windows, Mac OS* y *Linux*; despliega un único desarrollo en 25 plataformas diferentes. “Tiene soporte para mapeado de relieve, mapeado de reflejos, mapeado por paralaje, oclusión ambiental en espacio de pantalla, sombras dinámicas utilizando mapas de sombras, *render* de texturas y efectos de post-procesamiento de imagen” (UNITY, 2019).

Figura 14: Logo de unity sobre una escena 3D

Destacamos algunas imágenes de videojuegos populares que se han desarrollado con *Untity*:

Figura 15: Dexeus Ex: The Fall (2013)

Figura 16: Assassin's Creed: Identity (2014)

Figura 17: Satellite Reign (2014)

Figura 18: Kerbal Space Program (2011)

4.1.1 Scripting

Aunque Unity utiliza una implementación de un tiempo de ejecución estándar Mono para scripting, éste tiene sus propias prácticas para acceder al motor desde *scripts*. El comportamiento de los objetos, denominados *GameObjects*, es controlado por los componentes que lleva adjuntos. Aunque estos componentes pueden servirnos para muchas

funciones no son suficientes y vamos a necesitar implementar otras características añadiendo *scripts* con carácter de componente.
Unity soporta dos lenguajes de manera nativa:

- C# (C-sharp), un lenguaje estándar similar a Java o C++;
- UnityScript, un lenguaje diseñado específicamente para Unity y modelado detrás de JavaScript;

Cuando se crea un objeto y se le añade un script como componente aparecen por defecto dos funciones definidas dentro de la clase:

- La función *Update* es donde debe colocarse el código que se encargará de actualizar el *GameObject* en cada *frame*.
- La función *Start* va a ser llamada por Unity antes de empezar a interactuar con la app y antes de que la función *Update* sea llamada por la primera vez. Es donde debe incluirse cualquier tipo de inicialización.

El aspecto inicial de un *script* por defecto de Unity es el siguiente:

```
using UnityEngine;
using System.Collections;

public class MainPlayer : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
}
```

Figura 19: Script de Unity sin editar.

Cuando creamos un *script*, esencialmente estamos creando un nuevo tipo de componente que puede ser adjuntado a *GameObjects* como cualquier otro componente. Por lo tanto si no lo enlazamos a un objeto no está instanciado y por lo tanto es como si no existiera.

4.2 Herramientas AR para desarrollo software (*SDK*)

Los *SDK* (Software Development Kit) suelen incluir un entorno de desarrollo integrado (*IDE*), que sirve como interfaz de programación. El *IDE* puede incluir un editor de texto para escribir el código fuente, un depurador para corregir errores del programa y un editor visual, que permite a los desarrolladores crear y editar la interfaz gráfica de usuario del programa. Los *IDE* también incluyen un compilador, que se utiliza para crear aplicaciones a partir de archivos de código fuente.

En 2017 *Apple* y *Google* lanzaron sus propios kit de desarrollo de realidad aumentada, *ARKIT* y *ARCore*.

A continuación analizaremos ambas tecnologías, realizaremos una comparativa a partir de la información de su web, y nos decidiremos por una de las dos para este proyecto.

4.2.1 ARKit

ARKit Es la plataforma de desarrollo de realidad aumentada de *Apple* para dispositivos móviles *iOS*. Se introdujo junto con *iOS* 11 y está especificado para ejecutarse en *Core A9* y superiores.

Puntos clave:

- Seguimiento (*tracking*): dispone de la capacidad de rastrear con precisión la posición del dispositivo en el mundo real. Usando el cuentakilómetros visual inercial (*VIO*) combina *ARKit* con el seguimiento de la cámara y el sensor de movimiento, y da con la posición real exacta del dispositivo.
- Comprensión del paisaje y percepción de los rayos: los dispositivos *iOS* son especialmente conscientes del entorno gracias a *ARKit*. Dispone de la capacidad de identificar superficies del mundo real, como el suelo, mesas, paredes, etc. También se conocido como "*Detección de planos*".
- Renderizado: proporciona una fácil integración con SpriteKit, SceneKit, con un soporte adicional para *Unity* y *Unreal Engine.*

4.2.2 ARCore

ARCore es un kit de desarrollo de software desarrollado por Google el 1 de marzo de 2018 para Android 7.0 y superiores.

Puntos clave:

- Rastreo de movimiento: ARCore recopila los datos de la unidad de medición inercial y los puntos característicos del espacio de alrededor para determinar dinámicamente tanto la posición como la orientación del dispositivo.

- Comprensión ambiental: ARCore detecta superficies horizontales utilizando métodos similares a los de rastreo de movimiento.
- Estimación de la luz: ARCore detecta el ambiente de iluminación del dispositivo, mejorando así la apariencia y haciendo que la imagen se adapte al ambiente en tiempo real.
- Interacción del usuario: con la función "*hit-testing*" *(prueba de impacto)*, ARCore detecta la intersección de los rayos de luz en la vista de la cámara del dispositivo
- Anclaje de objetos: para colocar con precisión un objeto virtual, ARCore define un ancla que asegura su capacidad de rastrear el desplazamiento del objeto durante un período de tiempo. ARCore mejora eficientemente su comprensión de la posición y el entorno.

4.2.3 ARCore vs. ARKit

Recordamos que esta decisión entre ambas tecnologías ha sido la fase del proyecto que más demora ha generado respecto a la planificación de tareas inicial. Mientras *ARKit* nos permitía desarrollar la idea inicial de dibujar un patrón simple en la piel aproximado al *tracker* original, es decir, presentaba mejor tecnología de detección y seguimiento; *ARCore* necesitaba un patrón más complejo e idéntico al original.

Por otro lado sabemos que *ARKit* de Apple está restringido a dispositivos iOS (con iOS 11), mientras que *ARCore* de Google está disponible en *Android 7.0* o superior, como el *smartphone* del que disponemos.

Tuvimos que realizar numerosas pruebas con diferentes patrones para asegurarnos que las dos tecnologías de seguimiento tenían una lógica completamente distinta.

Analicemos qué diferencias las separan:

- Con respecto a los **dispositivos compatibles**, como se ha mencionado anteriormente, el ARKit de Apple Inc. está disponible solo para iPhones y iPads con iOS 11. En cambio, Google también planea introducir ARCore en la Web presentando un navegador web que permita a los desarrolladores crear sitios web habilitados para usar AR.
- En cuanto a los **requisitos de software**, tanto ARKit como ARCore son bastante similares. Ambos trabajan con Java/OpenGL, Unity y Unreal y utilizan la comprensión ambiental, sensores de movimiento y detección de luz.
- Sin embargo, una de las áreas en las que ARCore tiene una clara ventaja es en el **mapeo (*mapping*)**. El mapeo es la capacidad de recopilar y almacenar información de localización sobre el mundo real en tres dimensiones. ARKit utiliza una "*ventana*

deslizante" que solo almacena una cantidad limitada de datos temporales. Por lo tanto, ARCore tiene la capacidad de administrar datos de mapas mucho más grandes que le proporciona un esquema de datos más estable.

- En términos de **exposición en el mercado** se refiere, encontramos estas dos referencias de páginas webs de análisis tecnológico:
- "Google tiene una ligera ventaja ya que ha incursionado en áreas similares con sus experimentos anteriores con *Project Tango*[2] y *Google Cardboard*[3]. Sin embargo, para Apple, esta es su primera gran incursión en esta área" (Newgenapps, 2017).

  "Curiosamente, ARCore tiene más similitudes con ARKit de Apple que con Tango, ya que no requiere hardware adicional y funciona con una amplia variedad de dispositivos por medio de software" (Xataka, 2017).

Realizando pruebas con ambas tecnologías para el uso que necesitamos darle y sopesando las ventajas e inconvenientes de cada una de ellas, nos decidimos por ARCore de Google.
En primer lugar, lo que más nos importa en este caso es el material del que disponemos. Si no contamos con dispositivos iOS para probarlo, ni un Mac para compilarlo, el presupuesto del proyecto se dispararía considerablemente así que decidimos usar ARCore con dispositivos Android y compilando con Windows 10.
En segundo lugar, el mejor mapeo de ARCore nos repercute directamente en el resultado de la aplicación del tatuaje. Probando con ARCore el diseño permanece firme sobre el *tracker* y, al perderlo de vista con la cámara, el sistema conoce en todo momento la posición en el entorno, y al volverlo a enfocar sigue en el lugar donde debería estar.
En cambio *ARKit*, además de generar más vibración e inestabilidad del diseño aplicado, es capaz de perderlo y olvidar su localización en algunos casos.

4.2.4 AR Foundation

AR Foundation es un marco de trabajo de *Unity* creado especialmente para el desarrollo AR que nos permite crear proyectos de realidad enriquecida y luego exportarlas en diferentes dispositivos AR móviles. En otras palabras es un *framework* que busca unir las mejores capacidades de *ARCore* y de *ARKit*, con el fin de trabajar independientemente del destino final.

[2] Project Tango es una herramienta portátil de visión artificial que permite mapear espacios 3D con Phab2 Pro de Lenovo. El soporte a esta tecnología terminó el pasado 1 de marzo de 2018.
[3] Google Cardboard es una plataforma de realidad virtual en forma de gafas a base de cartón plegable, que funciona a partir de montar un teléfono móvil inteligente con Android o iOS en su interior.

Por lo tanto la decisión anterior (entre *ARKit* y *ARCore*) no nos afecta tanto a nivel de código sino como a la puesta en práctica. A continuación mostramos un cuadro compartido por Unity que compara las propiedades que ofrece cada tecnología:

Supported Feature	AR Foundation	Google ARCore SDK for Unity	Unity ARKit Plugin
Plane Detection (Vertical)	✔	✔	✔
Plane Detection (Horizontal)	✔	✔	✔
Feature Point Detection	✔	✔ + Oriented Feature Points	✔
Light Estimation	✔	✔ + Color Correction	✔ + Color Temperature
Hit Testing (Feature point and Plane raycasting)	✔	✔	✔
Image Tracking	in development	✔ (Static Only)	✔
3D Object Tracking	in development		✔
Environment Probes	in development		✔
World Maps	✔		✔
Face Tracking (Pose, Mesh, Blendshapes)	✔		✔ iPhone X + Variants Only
Cloud Anchors	in development	✔	
Editor Remoting	in preview	✔ - Instant Preview	✔ - ARKit Remote
Editor Simulation	in preview		
LWRP support (+ Shader Graph*)	3.3.0 supported	in development	in development
Camera Image API	✔		

Figura 20: Comparativa entre AR Foundation, ARCore, y ARKit

5 Desarrollo

5.1 Funcionalidades generales

Aunque a la larga se acaben desarrollando más funcionalidades y herramientas, hay algunas de ellas que se consideran fundamentales para la usabilidad de la aplicación.
Como funcionalidades principales destacamos:

- **Galería de diseños:** Consistirá en una lista deslizante horizontal, con los diseños en el interior de máscaras circulares. El diseño seleccionado es el que coincide con el botón de captura con forma de aro central.
- **Subir diseños en galería propia:** Mediante un botón de adición (+) añadiríamos a la galería anterior un diseño de nuestra galería de imágenes personal.
- **Navegar en la galería de imágenes:** Nos permite movernos por nuestra galería de imágenes, previsualizar, descartar o seleccionar diseños.
- **Función para capturar una vez el *marker* se ha reconocido:** Una vez el *marker* se ha reconocido, debe permitir capturar imagen o video del tatuaje.
- **Guardar las capturas en la galería del usuario:** Si el material visual capturado es el deseado, puede guardarse en la galería del usuario, o descartarlo de lo contrario.

5.2 Funcionalidades específicas

A continuación listamos de manera más específica y detallada qué funciones planificamos que tendrá la aplicación. En color gris oscuro se muestran aquellas que no hemos implementado. Como se explica al final de esta sección, o bien porque las hemos sustituido por otras, o no ha dado tiempo de desarrollarlas y quedan como futuras mejoras.

1 Listado de diseños

1.1 Deslizar diseños por defecto en *scroll* horizontal
1.2 Seleccionar diseño del listado
1.3 Añadir diseño al listado desde la galería propia
1.4 Eliminar diseño propio del listado

2 Galería propia

2.1 Listar galería propia
2.2 Seleccionar diseño
2.3 Cerrar galería propia

2.4 Eliminación del fondo de la imagen

3 **Visualización del diseño propio**

3.1 Abrir modal

3.2 Seleccionar la imagen

3.3 descartar la imagen (cerrar modal)

4 Pantalla de ayuda

4.1 Abrir instrucciones textuales

4.2 Cerrar instrucciones textuales

4.3 Vídeo tutorial

5 Emplazar y capturar contenido

5.1 Detección del *tracker* e instanciación del contenido (AR Foundation)

5.2 Eliminación del *tracker* una vez se ha instanciado el contenido

5.3 Escalado del contenido

5.4 Rotación del contenido

5.5 Captura de vídeo de la pantalla mientras se pulsa el botón de captura

5.6 Captura de imagen de la pantalla

5.7 Fusión del color y transparencia del contenido con la imagen de la cámara.

5.8 Superposición del *tracker* en la pantalla como una plantilla para delimitar una área de detección

5.9 Eliminar área del diseño que sobresalga del límite del cuerpo

6 Revisar captura

6.1 Previsualización en bucle del vídeo grabado.

6.2 Previsualización de la imagen capturada.

6.3 Descartar diseño.

6.4 Guardar diseño.

Analicemos los puntos grises que son aquellos que no han sido añadidos todavía:

- El 4.3 consiste en un vídeo-tutorial con el proceso de manejo de la aplicación. No ha sido añadido todavía por falta de tiempo.
- El 5.2 se refiere a eliminar el dibujo del adhesivo, es decir, tras el tatuaje proyectado, si este no es lo suficientemente oscuro, se puede ver el diseño del tracker en el adhesivo. Esta función debería encargarse de eliminarlo para una proyección más realista del

tatuaje. De momento no hemos implementado esta función pero hemos planteado soluciones.

1. La primera de ellas, la más simple, consiste en reducir la parte negra del tracker del adhesivo, por ejemplo:

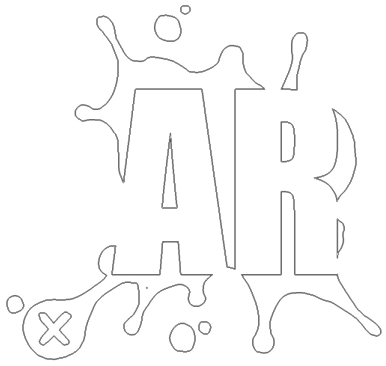

Figura 21: Logo contorneado simplificado

2. La segunda consiste en proyectar un plano color carne por debajo de la proyección del tatuaje, con el *shader* adecuado, debería disimularse con el tono de la piel y quitarle opacidad al adhesivo.
3. Por último, aunque puede haber más alternativas, consistiría en el postprocesado de la imagen que recibe la cámara y se encargaría de copiar los píxeles cercanos a los que están ocupados por el tracker para eliminarlo.

- El 5.8 se refiere a crear una ventana cuadrada semitransparente, superpuesta a la cámara para delimitar un área de detección y que el usuario enfocase con la distancia correcta entre el dispositivo móvil y el adhesivo. Consideramos que este punto no es necesario porque el sistema AR es tan sólido que instancia rápidamente el plano sin necesidad de esforzarse en captarlo.
- El 5.9 es un tema más tedioso, se debe analizar y procesar la imagen que está recibiendo la cámara, poder analizar qué es parte del cuerpo y qué es fondo; entonces relacionar esta información con el recorte del plano que se está proyectando. Este es uno de los puntos con más profundidad de estudio para futuras mejoras.

5.3 Diagrama de flujo

Uno de los objetivos principales de esta aplicación es ser simple y agradable. Por ese motivo su funcionamiento es muy básico e intuitivo. A continuación mostramos un diagrama del flujo de la aplicación:

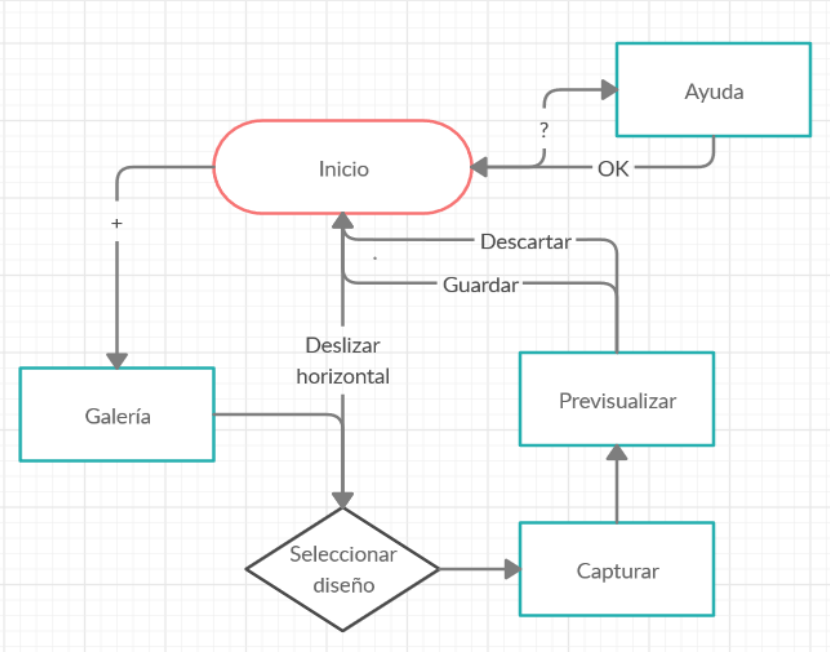

Figura 22: Diagrama de flujo de TattooARt

5.4 Diseño

5.4.1 Elementos de diseño

En un primer momento planteamos un diseño un poco más rudimentario, con más botones e interfaces que el actual.

Analizando cuáles son las aplicaciones AR que más atraen al público entre 18 y 35 años, que es donde se concentra el mayor volumen potencial de clientes del tatuaje, destacamos el uso de *Instagram* y *Snapchat*, en especial este último, que sigue un flujo muy inspirador para esta aplicación.

Figura 23: Icono de escritorio de TattooARt sobre fondo aleatorio

En primer lugar encontramos en nuestra pantalla, junto a otras aplicaciones, un logo atractivo: dos letras en blanco AR (*Augmented Reality*) tras un fondo de una mancha de tinta negra, como mueca hacia el tema que trata. El logo está extraído del título y del patrón que se repartirá en pegatinas. Generando así una coherencia visual entre los estilos de diseño.

El hecho de abrir la aplicación y encontrarse directamente con la cámara y un listado de diseños que puede desplazar inmediatamente, incita al usuario a realizar una acción, a ser rápido, práctico y dar con su objetivo. Si la finalidad de esta solución informática es ahorrar tiempo en un proceso muy claro, no podemos obligar al usuario a navegar por pantallas o a seguir un flujo complejo de acciones.

Es por eso que decidimos este diseño, siguiendo un esquema sencillo, y una paleta de colores muy contrastados: blanco, negro y rojo.

A continuación mostramos algunas capturas de la escena principal de la aplicación, así como algunos de sus botones diseñados con *Photoshop CC 2017:*

Figura 24: Parte inferior (selector de diseños, botón de añadir y botón de borrar)

Figura 25: Parte superior (título y botón de ayuda)

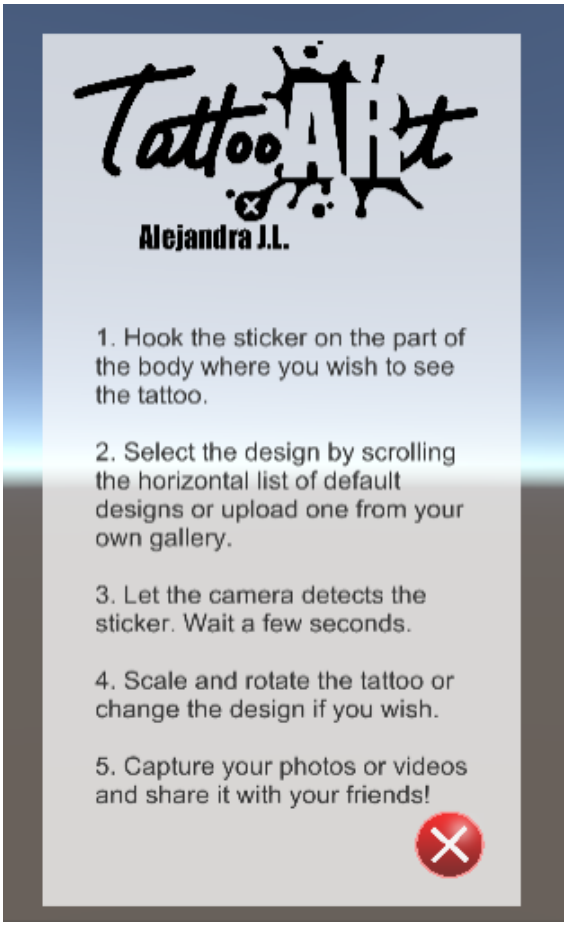

Figura 26: Pantalla de ayuda y botón para cerrarla

Figura 27: Botón de añadir

Figura 28: Botón de ayuda

Figura 29: Botón de borrar.

5.4.2 Wireframes y flujo de la aplicación

Con los elementos anteriormente descritos, el sistema de pantallas de la aplicación queda como en la figura siguiente:

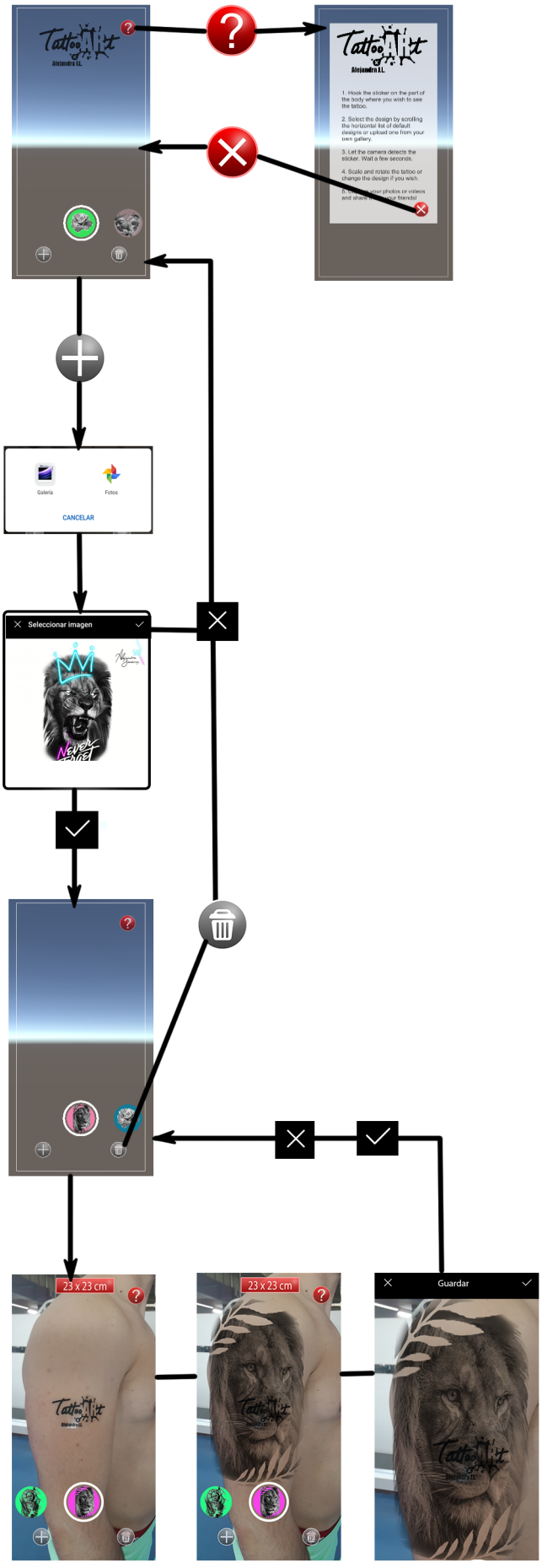

Figura 30: Flujo de la aplicación con wireframes

5.5 Patrón de seguimiento AR

Esta fue una etapa costosa debido a que en un primer momento no conocíamos el sistema de generación del mapa de características de *ARFoundation* para obtener información del tracker. Así que no cualquier tracker era válido ni obteníamos un buen resultado de detección.
Encontramos una herramienta ejecutable para la consola que podías referenciarle modelos de patrones y generaba una puntuación del 1 al 100 la generación de características de este. Realizando múltiples pruebas pudimos observar que la intersección de líneas u otras formas, y la proximidad de los contornos en el interior del patrón generaba mejores resultados.
Finalmente dimos con la combinación entre consistencia, diseño y publicidad; a continuación mostramos la que será la imagen del patrón:

Figura 31: Patrón de seguimiento o tracker

5.5.1 Impresión de adhesivos

Después de analizar dónde podíamos encargar los adhesivos, encontramos que vía online suponía una compra al por mayor demasiado grande para una primera versión. Así que decidimos obtener un número menor en una papelería local de confianza, con un precio de 2,50€ por hoja, donde pueden imprimirse 10 adhesivos.

Los adhesivos son transparente con el tracker en negro. Se puede desarrollar una versión en blanco para usuarios de color. Está testeado sobre diferentes fondos de piel y funciona correctamente.

En la segunda versión, con la intención de crear una imagen más sólida de la marca y estrechando la relación entre la aplicación y el diseñador/tatuador, se ha añadido el número de contacto para reservar cita y la página de *Instagram* donde se puede consultar el trabajo del artista.

Figura 32: Patrón con información de contacto para los adhesivos

5.6 Implementación

La implementación se ha desarrollado en Unity, que permite añadir visualmente los elementos y relacionarlos simplemente arrastrándolos; todo lo que no puede llegar a relacionarse con este modo se hace mediante scripting, en este caso en el lenguaje C#.
Ha supuesto un esfuerzo mayor por el desconocimiento de este lenguaje ni de esta plataforma pero también nos ha servido para aprender una parte de su estructura y funcionamiento.
Mostraremos a continuación algunos de los puntos de la implementación más importantes, centrándonos en las funciones más relevantes e interesantes a nivel implementación.

5.6.1 Implementación del listado de diseños

El listado de diseños pretende imitar un sistema de Scroll muy popular recientemente. Consiste en una lista de paneles circulares que contienen como textura un diseño. Se puede desplazar horizontalmente y se considera seleccionado el que se queda en el interior del círculo. A continuación adjuntamos una comparativa entre el *scroll* de *Snapchat*, *Instagram* y *TattooARt*:

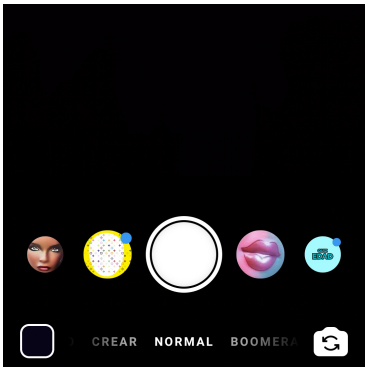

Figura 33: Scroll de Snapchat

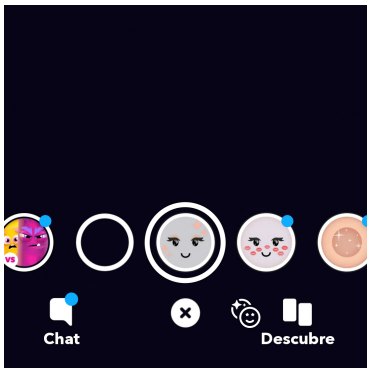

Figura 34: Scroll de Instagram

Figura 35: Scroll de TatooARt

Este sistema de selección está respaldado por un *Asset* llamado S*imple Scroll Snap,* que nos brinda algunas facilidades y herramientas para mejorar el sistema de *scrolling* y selección. Como, por ejemplo: la aceleración de los paneles, el desplazamiento automático hacia el centro cuando queda próximo al circulo, y algunas propiedades más. Los botones circulares son objetos definidos por una máscara circular y una imagen en su interior. Incluimos una serie de diseños por defecto que ya están añadidos automáticamente cuando se inicia el *Game Play.* Mediante el símbolo de "+" se añade un nuevo panel en el inicio del listado que se obtiene de la galería del usuario. Entonces la textura de galería debemos adjudicarla al botón. Con el botón de la papelera eliminamos este último que acabamos de añadir. No se pueden borrar los paneles que están por defecto para proteger el sistema.

A continuación mostramos unas lineas de código de cómo añadimos y como borramos un panel:

```
private void Add(int index)
{
    //Pagination
    float toggleWidth = toggle.GetComponent<RectTransform>().sizeDelta.x;
    Instantiate(toggle, sss.pagination.transform.position +
        new Vector3(toggleWidth * (sss.NumberOfPanels + 1), 0, 0),
        Quaternion.identity, sss.pagination.transform);
    sss.pagination.transform.position -= new Vector3(toggleWidth / 2f, 0, 0);

    //Panel
    panel.GetComponent<Image>().color =
        new Color(UnityEngine.Random.Range(0, 255) / 255f,
        UnityEngine.Random.Range(0, 255) / 255f,
private void Remove(int index)
{
    if (sss.NumberOfPanels > 0 & (sss.NumberOfPanels-index)>4)
    {
        //Pagination
        DestroyImmediate(sss.pagination.transform.GetChild(sss.NumberOfPanels - 1).gameObject);
        sss.pagination.transform.position += new Vector3(toggleWidth / 2f, 0, 0);
        //Panel
        sss.Remove(index);
    }
}
```

Figura 37: Código de borrar panel

5.6.2 Implementación de captura y acceso a galería

Permitir el acceso a galería es un tema interesante, sobretodo porque nos encontramos con que iOs opone más dificultades que Android para poder acceder a la galería personal del usuario. En este caso para poder acceder, cuando seleccionamos el botón de añadir panel por primera vez, necesitamos concederle los permisos:

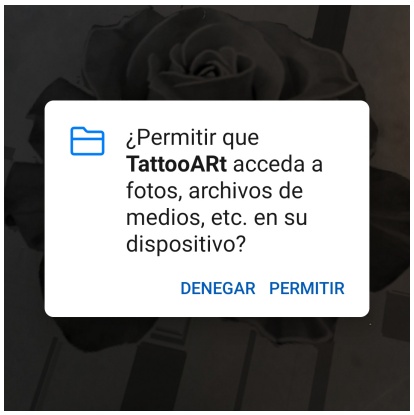

Figura 38: Pop-up de permisos a galería

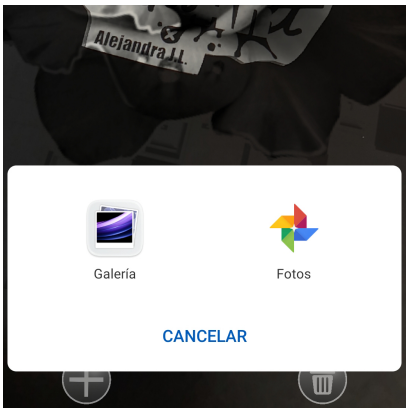

Figura 39: Pop-up de selección de galería

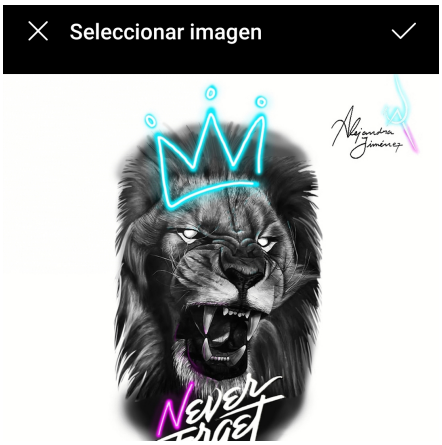

Figura 40: Selección de diseño

Para poder implementar estos accesos nos hemos servido de un *asset* llamado *Native Gallery,* que permite pedir los permisos y acceder a galería tanto para subir o bajar contenido de ella. A continuación adjuntamos un fragmento de código de cómo accedemos a galería:

```
public void Selectimage(int maxSize)
{
    NativeGallery.Permission permission = NativeGallery.GetImageFromGallery((path) =>
    {
        Debug.Log("Image path: " + path);
        if (path != null)
        {
            Texture2D tempTexture = NativeGallery.LoadImageAtPath(path, maxSize);
            if (tempTexture == null)
            {
                Debug.Log("Couldn't load texture from " + path);
            }
            else
            {
                Debug.Log("Loaded texture from " + path);
                tattooMaterial.SetTexture("_MainTex", tempTexture);
            }
        }
    }, "Select a PNG image", "image/png");
    Debug.Log("Permission result: " + permission);
}
```

Figura 41: Código para seleccionar una imagen de Galería

Para poder capturar el contenido y guardarlo en la galería, se ha tenido que controlar el pulsado del botón central, pues no es lo mismo un toque, que representa realizar una instantánea o, mantener el botón pulsado que representa grabar un vídeo, tal y como estamos acostumbrados en la mayoría de aplicaciones. Para esto existen funciones, como *GetButtonDown(),* relacionadas con los Objetos *Button* que permiten acceder al tipo de interacción que está recibiendo.

5.6.3 Implementación de detección e instanciación de contenido AR

Esta es la parte más importante ya que no es común a todas las aplicaciones como puede notarse en las secciones anteriores. Para poder detectar un patrón e instanciar un contenido, que es el objetivo intrínseco de esta aplicación móvil, necesitamos la plataforma anteriormente mencionada: AR Foundation.

Una jerarquía de escena AR básica se ve así:

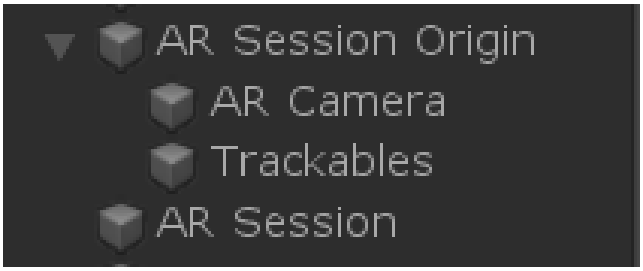

Figura 42: Jerarquía de una sesión AR

Tal y como explica la página de Unity(2019) sobre estas dos sesiones por defecto:

> "Una escena AR debe incluir un componente *ARSession*. La sesión AR controla el ciclo de vida de una experiencia AR, habilitando o deshabilitando AR en la plataforma de destino. El *ARSession* puede estar en cualquier *GameObject*."
>
> "El propósito de *ARSessionOrigin* es transformar características rastreables (como superficies planas y puntos) a su posición final, orientación y escala en la escena de Unity. Debido a que los dispositivos AR proporcionan sus datos en el "espacio de sesión", si tenemos un espacio sin escala al comienzo de la sesión AR, *ARSessionOrigin* realiza la transformación adecuada al espacio Unity."
>
> "El ARSessionOrigin también le permite escalar contenido virtual y aplicar un desplazamiento a la cámara. Dado que la cámara es controlada por la sesión, la cámara y los rastreables detectados se moverán juntos en esta nueva configuración."

Junto al '*GameObject de ARSessionOrigin*' debe haber, mínimo, una cámara, que se utilizará para renderizar cualquier *trackable* que desee visualizar. La cámara también debe tener un componente *TrackedPoseDriver*, que controlará la posición local y la rotación de la cámara de acuerdo con la información de seguimiento del dispositivo. Esta configuración permite que el espacio local de la cámara coincida con el "*session space*" de AR.

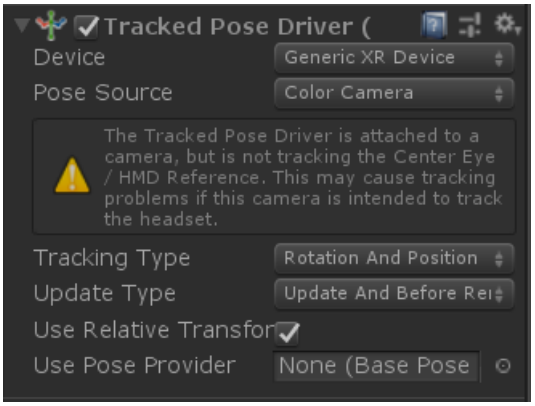

Figura 43: Componente de la cámara AR

Cuando la cámara detecta un *marker,* su *Manager* instancia un determinado *prefab,* en nuestro caso cuando se detecta el marcador añadido en *trackables* se instancia un plano que en su interior tiene este aspecto:

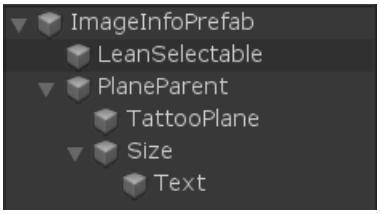

Figura 44: Prefab del plano

Podemos destacar dentro de este *Prefab* los siguientes elementos:

- **Lean Selectable:** es un *Asset* que permite que el plano instanciado pueda ser seleccionado con los dedos y así se pueda escalar y rotar de la forma en la que estamos acostumbrados, con dos dedos a modo de pinza.

- **TattooPlane:** es el plano que contiene el tatuaje en su material y se modificará cuando el diseño quiera ser cambiado por otro, cuando se escale o rote.

- **Size:** Es un elemento de texto que sirve de “chivato” para conocer las dimensiones del plano y poder imprimirlo en el tamaño correcto.

Y a continuación vemos información general del plano instanciado:

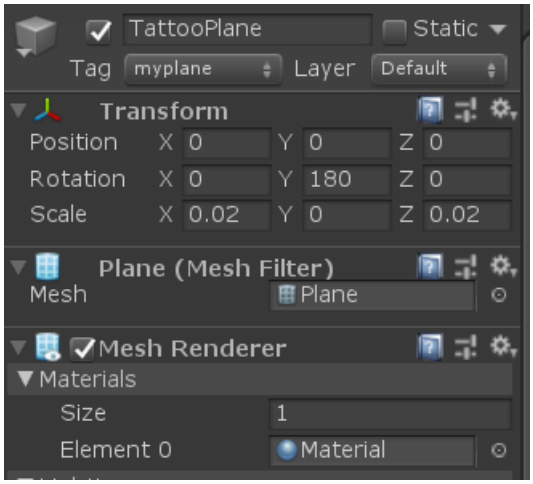

Figura 45: Inspector del plano que contiene el tatuaje

También es importante la característica de *Raycasting,* también conocido como *hit-testing*, la emisión de rayos le permite determinar dónde un rayo (definido por un origen y dirección) se cruza con un *marker*. La interfaz actual del *Raycast* solo prueba contra planos y puntos. La interfaz de emisión de rayos es similar a la del módulo Unity Physics, pero los marcadores AR de Unity no necesariamente tienen presencia en el mundo real, así que AR Foundation proporciona una interfaz separada.

No podemos capturar como se instancia el plano del tatuaje mediante nuestro escritorio porque solo funciona con la cámara de nuestro dispositivo móvil. Esto ha supuesto una gran demora para *debuguear*, ya que necesitábamos descargarnos e instalar la aplicación tras cada cambio importante en la implementación.

A continuación mostramos una captura de como queda instanciado un tatuaje una vez se ha detectado el marcador, en nuestro caso sobre el adhesivo que hemos diseñado como *marker*:

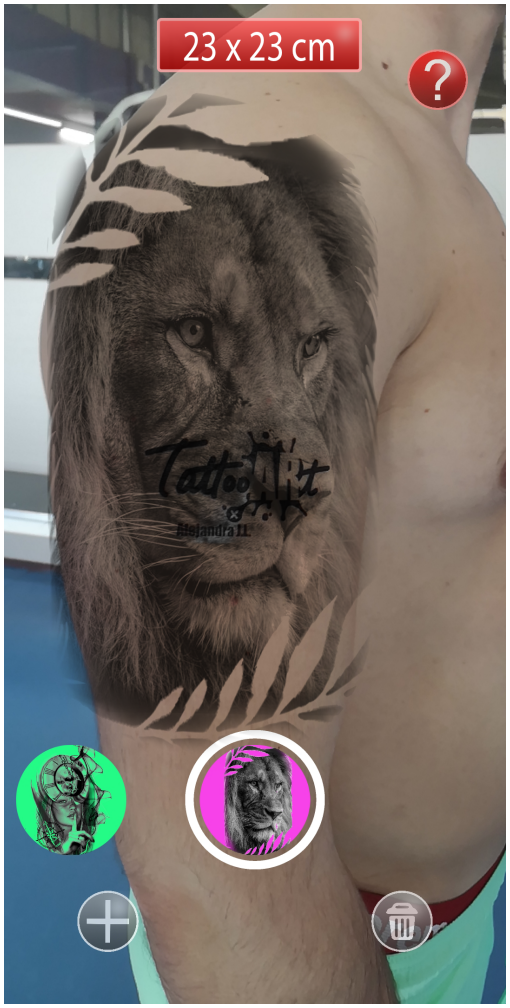

Figura 46: Imagen resultado de la instanciación del tatuaje sobre el adhesivo

6 Presupuesto y sostenibilidad

6.1 Presupuesto

En este apartado se va analizar el coste económico del proyecto. Los gastos estrictamente asociados a la realización del proyecto se dividen en tres partes: *hardware*, *software* y recursos humanos. Por otro lado distinguimos aquellos más generales o indirectos (alquiler del local, material utilizado, agua y luz), los imprevistos y las contingencias.

6.1.1 Costes de *Hardware*

En el apartado de coste de *Hardware* se tiene en consideración el equipo informático que se ha necesitado para el desarrollo del proyecto.

Hardware	**Especificaciones**	**Precio**
Portátil ASUS	● Core i7 8th Gen ● GTX 1600 ● 16 Gb	1800€
Huawei p20	● Android Oreo 8.1, EMUI 8.1. ● 6GB de RAM	360€
Router inalámbrico	● Velocidad 300 Mbps	30 €

Total	**2.190€**

Tabla 4: Tabla con los costes de Hardware

6.1.1.1 Control de gestión

Hay que tener en cuenta que lo gastado en *hardware* no tiene repercusión directa en el coste del proyecto. El conocido proceso de amortización hace que el material acabe suponiendo un gasto nulo. Considerando que el ordenador tardará 4 años en estar amortizado, el *Smartphone* 1 año, y el *router* 4 años, calculamos el valor imputable del *hardware* al proyecto respecto a la siguiente fórmula:

Precio del equipo € / (años de vida útil * 220 días laborables/año* X horas dedicación/día) y el resultado * X horas dedicación TFG

6.1.2 Costes de *Software*

Hay que valorar también el software que se ha utilizado durante el proyecto, aunque en este caso no necesitamos licencias de pago para ninguna plataforma de las que utilizamos.

Windows 10	**60€**
IDE Android Studio	**Gratis**
SQL Server Express	**Gratis**
Sistema operativo *Android*	**Gratis**
Repositorio de código	**Gratis**
ARToolkit de Desarrollo	**Gratis**
Licencia *UNITY*	**Gratis**
Total	**60€**

Tabla 5: Tabla con los costes de Software

6.1.3 Costes de recursos humanos

Para analizar estos costes hemos consultado a una empresa que forma parte en el crecimiento de este proyecto donde analistas, desarrolladores y diseñadores nos han podido dar una idea del sueldo esperado para cada posición.

Descripción	**Tiempo estimado**	**Profesional**	**precio/ hora**	**total**
Gestión del proyecto	**40h**	**analista**	**45**	**2.464**
Análisis de la competencia	**4h**			
Comparativa con la app destacada	**4h**			
Definición funcional	**4h**			
Diseño del esquema de la aplicación	**8h**	**diseñador**	**45**	**360**
Pruebas con diferentes	**8h**	**programador**	**35**	**980**

herramientas				
Pruebas con las tecnologías AR	**8h**			
Decidir la plataforma	**4h**			
Nivel de AR y diseño del patrón	**8h**			
Diseño	**16h**	**diseñador**	**45**	**720**
Programación del código	**32h**	**programador**	**35**	**2240**
Testeo	**12h**			
Corrección de bugs	**8h**			
Mejoras	**12h**			
Herramientas de seguimiento	**8h**			
Publicación	**4h**	**analista**	**45**	**900**
Seguimiento	**4h**			
Actualización	**12h**			
Total **seguridad social (35%)**	**7.664€ * 1,35 = 10.346,4 €**			

Tabla 6: Tabla con los costes de recursos humanos

6.1.4 Costes indirectos

Estos son aquellos costes relacionados con el entorno de trabajo, tanto de espacio como recursos materiales y consumibles.

Gasto indirecto	**Meses**	**€/mes**	**Total**
Alquiler del local	3 meses	1.000€	3.000€
Agua, gas y luz	3 meses	100€	300€
Material	3 meses	15€	45€
TOTAL		**3.345€**	

Tabla 7: Tabla con los costes indirectos

6.1.5 Contingencias

La contingencia, entendida como coste genérico para cubrir obstáculos no previstos. Depende del sector y del nivel de detalle en que se hace el presupuesto. En el caso del sector informático suelen fijarse ratios de entre el 10% y el 20%.

Suma de costes: 185,8 + 10.346,4 + 3.345 = **13,877.2**
Ratio de contingencia: **15%**

Total: 13,877.2 * 1,15= **15,958.78€**

6.1.6 Revisión del presupuesto

En el punto en el que se planteó que la tecnología de ARCore era inadecuada para el desarrollo del proyecto se discutió la posibilidad de cambiar a iOS. Este cambio generaba un incremento en el coste de *hardware* que no se podía amortizar en pocos años vista y por eso sucumbimos tomar la vía alternativa y decidirnos por los adhesivos con el patrón impreso.
Este nuevo añadido también supuso un incremento en la inversión y el presupuesto, pero muy inferior aún así al cambio de *hardware*.
Calculamos que para para obtener 400 adhesivos que nos servirían para cubrir el primer año de promoción de la aplicación deberíamos invertir 60€ en su impresión. Suponiendo un gasto total del proyecto de 16,018.78€.

6.2 Sostenibilidad

Realizar un análisis de la sostenibilidad social, ambiental y económica de tu proyecto es algo más complejo de lo que parece. Mientras que el impacto económico incluye los recursos humanos y materiales del proyecto y puedes medir con más exactitud en qué medida puedes reutilizar o minimizar este coste, no queda tan claro el impacto social y medioambiental que puedes generar.
Realizando el test de sostenibilidad y reflexionando sobre las preguntas que propone la universidad para el estudio del impacto ambiental, plantearía trabajar en horarios donde se pudiese aprovechar la luz solar, reutilizar material de segunda mano, reducir los plásticos en nuestras mesas de la oficina y reciclar todo lo que se deseche durante las horas de trabajo. Por lo que respecta a la utilidad de la aplicación, ayudaría a reducir en gran escala el número de

papeles de transferencia que se utilizan para la plantilla del tatuaje pero supondría la impresión de adhesivos en material plástico, así que deberemos cuidarnos de su reciclaje.

Relacionado con el impacto social y emocional, en primera persona este proyecto supone una realización personal, una idea que lleva mucho tiempo en mi cabeza, que necesito desarrollar y poder aplicar en una problemática que forma parte de mi día a día. A nivel general, desearía que se le diera el uso para el que está pensado, que pudiera ayudar a otros tantos en su pasión artística y laboral y que pudiera servir a otros muchos a decidir algo que formará parte de su piel toda la vida. Como cualquier otra aplicación de edición multimedia, corre el riesgo de que se utilice para contenido no deseado o violento, para escenas de acoso o desatendidas del control parental. Para ello, llegado el momento de su publicación y distribución, veremos las limitaciones legales que podemos aplicar para reducir en la máxima medida el uso inadecuado de la aplicación.

7 Conclusiones y proyección de futuro

La realización de este proyecto ha supuesto una realización a nivel personal. Es una idea que llevaba mucho tiempo en mi cabeza y deseaba que mi proyecto fuese enfocado hacia este ámbito.

El desconocimiento de las tecnologías utilizadas, además de no haber cursado prácticas en este sector, me generaba una sensación de distanciamiento a la informática a nivel experimental o funcional. Inicié esta carrera porque deseaba poder darle vida a ideas que pasaban continuamente por mi cabeza. A su vez siempre quise excusar que la informática no estaba tan distanciada del arte, que de hecho podían ir de la mano en mi vida profesional y personal. Seguramente no sea el proyecto más grande en el que he pensado, ni la idea más compleja, pero el hecho de poder disponer de algo tan útil para mi, que he creado yo desde cero, me da un gran impulso para no desanimarme y empezar a buscar oportunidades creativas en este sector.

Cuando entré en el mundo del tatuaje, con 16 años, me di cuenta que era muy joven en este círculo y que no podía competir con tatuadores consagrados que llevaban 30 años en el sector; pero es cierto que ellos no nacieron con un móvil en la mano, como nosotros los *milenials*, que en esta práctica somos, aún, una minoría. Empecé a pensar que era un mundo muy tradicional, donde los cambios tecnológicos se integraban muy lentamente y los instrumentos y métodos de trabajo eran un poco arcaicos. Pues no me parece coherente que la gran mayoría de sectores de consumo común estén completamente informatizados, que cada vez la gente acepte con más gratitud el uso de la tecnología, y que en cambio, 1 de cada 3 españoles lleve un tatuaje y prácticamente ninguno de ellos sepa de ningún método para probarse el diseño que tiene pensado tatuarse el próximo mes (teniendo en cuenta que es un producto que no se puede devolver).

Me inspiré en esta idea con el uso diario de aplicaciones de AR, últimamente tan de moda, como *Snapchat* o los filtros de *Instagram.* Continuamente veía a usuarios y amigos que ponían en sus caras y cuerpos todo tipo de indumentarias 3D, y en cambio para enseñarles como les quedaría el tatuaje que tanto les gusta, tenía que hacer maniobras con el PC, el Photoshop y la tableta gráfica.

Pero no solo es por esta necesidad del consumidor, si no la del tatuador en cuestión.

Mientras que un cliente estándar puede necesitar este servicio 5 o 6 veces en su vida; alguien que dedica su jornada laboral a esto lo utilizaría hasta 5 o 6 veces al día.

Puede que la falta de mecanización del trabajo previo es debida a que no comporta una fabricación en masa como muchas otras producciones en cadena, en las que la automatización del proceso de producción es clave para el rendimiento económico de la empresa. Como analizábamos en el punto 3.1.4, un estudio de tatuajes que, en un caso ideal, tiene 5 tatuadores, y cada uno de ellos realiza una media de 4 tatuajes al día, ahorrando 20 minutos gracias al uso de esta aplicación; supondría un ahorro diario de 6h y 40 minutos. Suficiente para empezar a creer que la idea puede conllevar un beneficio económico significante.

La proyección de futuro de este proyecto a corto plazo la tengo bastante clara: añadir algunas mejoras a la interfaz de la aplicación y usarlo con todos mis clientes, repartir adhesivos a modo de visita y que cada uno de ellos pueda probar la aplicación en su móvil. También permitir que pueda ser una herramienta útil para tatuadores amigos de confianza que trabajan o han trabajado conmigo; conocer experiencias sobre la usabilidad de la app siempre va a ayudar a mejorarla.

A largo plazo, pienso en una idea más ambiciosa, en que esta plataforma sea un módulo de una plataforma más grande, una red social específica para esta práctica, como lo es la cámara de *Instagram* a su propia plataforma. Planteo una comunidad de usuarios en el que cada uno pueda tener su galería de diseños y su galería de capturas, y como usuario estándar puedes seguir a usuarios tatuadores, probarte sus diseños, incluso contactar con el.

8 Bibliografía

8.1 Libros y documentos

Greg Kipper, Joseph Rampolla. Augmented Reality: An Emerging Technologies Guide to AR. Syngress Media, 2013

Sanni Siltanen, Theory and applications of marker-based augmented reality. Finland, 2012

8.2 Web

UNITY, S.A. UNITY [en línia].[Consulta: 18 septiembre 2019]. Disponible en: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@2.2/manual/index.html

VUFORIA, S.A. VUFORIA [en línia].[Consulta: 18 septiembre 2019]. Disponible en: https://library.vuforia.com/articles/Training/getting-started-with-vuforia-in-unity.html

LEARNOPENGL, S.A. LEARNOPENGL [en línia].[Consulta: 18 septiembre 2019]. Disponible en:
https://learnopengl.com/Advanced-OpenGL/Blending

DEEPSKYCOLORS, S.A. DEESPKYCOLORS [en línia].[Consulta: 18 septiembre 2019]. Disponible en:
https://www.deepskycolors.com/archive/2010/04/21/formulas-for-Photoshop-blending-modes.html

NEW GEN APPS [en línea]. [Consulta: 2 diciembre 2019]
https://www.newgenapps.com/blog/arkit-vs-arcore-the-key-differences

8.3 Software utilizado

UNITY 3D. [Versión personal y gratuita] Disponible en:
https://unity3d.com

AR FOUNDATION [Versión gratuita] Disponible en:
https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@1.0/manual/index.html

ADOBE PHOTOSHOP [Versión de pago] Disponible en:
https://www.adobe.com/products/photoshop.html

MICROSOFT VISUAL STUDIO 2019 [Versión gratuita] Disponible en: https://visualstudio.microsoft.com/

9 Glosario

- **Tracking:** La capacidad del dispositivo AR para determinar su posición y orientación relativas en el mundo. Si el entorno es demasiado oscuro, por ejemplo, el dispositivo puede "perder el tracking", lo que significa que ya no puede informar con precisión su posición.
- **Trackable:** Una característica del mundo real detectada y / o rastreada por el dispositivo AR, por ejemplo, una superficie plana.
- **Feature Point:** Un punto específico en una nube de puntos. Los dispositivos AR utilizan una cámara y un análisis de imágenes para rastrear puntos específicos en el mundo que se utilizan para construir un mapa de su entorno. Estos son generalmente elementos de alta frecuencia, como un nudo en una superficie de grano de madera.
- **Asset:** Es la representación de cualquier elemento que pueda usarse en su juego o proyecto. Un asset puede provenir de un archivo creado fuera de Unity, como un modelo 3D, un archivo de audio, una imagen o cualquiera de los otros tipos de archivos que admite Unity.
- **Tracker/ patrón/ marker:** Los marcadores en realidad aumentada son señales visuales que activan la visualización de la información virtual. Los marcadores son imágenes normales u objetos pequeños que se analizan previamente y que el sistema entrena para que puedan reconocerse más tarde mediante la cámara.

  Después de que se reconoce un marcador, su posición, escala y rotación se derivan a señales visuales y se transfieren a la información virtual. **:**
- **Prefab:** Un Prefab de Unity le permite crear, configurar y almacenar un *GameObject* completo con todos sus componentes, propiedades, valores y *GameObjects* secundarios. Actúa como una plantilla a partir de la cual se pueden crear nuevas instancias prefabricadas en la escena.
- **Rendering:** Es el proceso automático de generar una imagen fotorrealista o no fotorrealista a partir de un modelo 2D o 3D (o modelos del archivo de escena) por medio de programas informáticos.